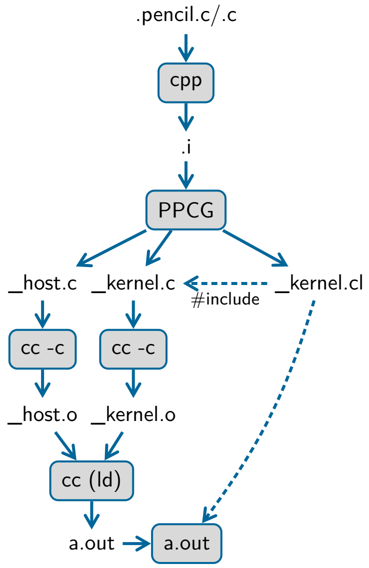
PENCIL

Practice Session with PPCG and PENCIL

Michael Kruse, Sven Verdoolaege

11th May 2016

`pencilcc` Driver



Requirements, Compile and Install

Requirements:

- gcc/clang
- GNU Autotools
- LLVM/Clang libraries
- Python 3
- OpenCL/CUDA in system default paths
- Ubuntu: `apt-get install build-essential autoconf libtool pkg-config llvm-dev libclang-dev libgmp-dev python3 opencl-headers clinfo nvidia-cuda-toolkit`

```
$ git clone https://github.com/Meinersbur/pencilcc.git --recursive
$ cd pencilcc
$ ./autogen.sh
$ ./configure --with-int=imath-32
$ make
$ make check
$ sudo make install
$ sudo ldconfig
```

Get the latest version

- Update sources

```
$ git fetch -all
$ git checkout origin/master
$ git submodule update --recursive
$ make
$ sudo make install
```

- Source example in `examples/practical/practical.c`

Running Example

- Data generation (sine + cosine waves)
- Gauss filter (convolution)
- Histogram
- Print histogram

gcc: Compile, Link, Run

- Compile

```
$ gcc -O3 practical.c -c -o practical.o
```

gcc: Compile, Link, Run

- Compile

```
$ gcc -O3 practical.c -c -o practical.o
```

- Compile and link

```
$ gcc -O3 practical.c -o practical
```

gcc: Compile, Link, Run

- Compile

```
$ gcc -O3 practical.c -c -o practical.o
```

- Compile and link

```
$ gcc -O3 practical.c -o practical
```

- Compile, link and run

```
$ gcc -O3 practical.c -o practical && ./practical
```

gcc: Compile, Link, Run

- Compile

```
$ gcc -O3 practical.c -c -o practical.o
```

- Compile and link

```
$ gcc -O3 practical.c -o practical
```

- Compile, link and run

```
$ gcc -O3 practical.c -o practical && ./practical
```

- Compile, link, run and measure execution time

```
$ gcc -O3 practical.c -o practical && time ./practical
```

pencilcc: Compile, Link, Run

- Compile

```
$ pencilcc -O3 --target=opencl practical.c -c -o practical.o
```

- Compilation taking too much time?

```
--no-private-memory
```

pencilcc: Compile, Link, Run

- Compile

```
$ pencilcc -O3 --target=opencl practical.c -c -o practical.o
```

- Compile and link

```
$ pencilcc -O3 --target=opencl practical.c -o practical
```

- Compilation taking too much time?

```
--no-private-memory
```

pencilcc: Compile, Link, Run

- Compile

```
$ pencilcc -O3 --target=opencl practical.c -c -o practical.o
```

- Compile and link

```
$ pencilcc -O3 --target=opencl practical.c -o practical
```

- Compile, link and run

```
$ pencilcc -O3 --target=opencl practical.c --run
```

- Compilation taking too much time?

```
--no-private-memory
```

pencilcc: Compile, Link, Run

- Compile

```
$ pencilcc -O3 --target=opencl practical.c -c -o practical.o
```

- Compile and link

```
$ pencilcc -O3 --target=opencl practical.c -o practical
```

- Compile, link and run

```
$ pencilcc -O3 --target=opencl practical.c --run
```

- Compile, link, run and measure execution time

```
$ pencilcc -O3 --target=opencl practical.c -o practical
$ time ./practical
```

- Compilation taking too much time?

```
--no-private-memory
```

Command Line Options

- Show (most) available options

```
$ pencilcc --help
$ ppcg --help
```

Command Line Options

- Show (most) available options

```
$ pencilcc --help
$ ppcg --help
```

- Verbose output

```
$ pencilcc -v
$ pencilcc -v -v
```

Command Line Options

- Show (most) available options

```
$ pencilcc --help
$ ppcg --help
```

- Verbose output

```
$ pencilcc -v
$ pencilcc -v -v
```

- Save intermediate files (`.i`, `_ppcg.c`, `_ppcg_kernel.c`, `_ppcg_kernel.cl`, `_host.o`, `_kernel.o`, `a.out`)

```
$ pencilcc --keep
```

Command Line Options

- Show (most) available options

```
$ pencilcc --help
$ ppcg --help
```

- Verbose output

```
$ pencilcc -v
$ pencilcc -v -v
```

- Save intermediate files (`.i, _ppcg.c, _ppcg_kernel.c, _ppcg_kernel.cl, _host.o, _kernel.o, a.out`)

```
$ pencilcc --keep
```

- File extensions to process

```
$ pencilcc --c-ext=.c
$ pencilcc --pencil-ext=.pencil.c
```

PRL Runtime Library (PRL)

- Use PRL

```
$ pencilcc --target=prl practical.c -o practical && ./practical
```

- Missing library `libprl_opencl.so` when launching?

```
--autorpath
```

PRL Runtime Library (PRL)

- Use PRL

```
$ pencilcc --target=prl practical.c -o practical && ./practical
```

- Select device to run on

```
$ PRL_TARGET_DEVICE=gpu ./practical
$ PRL_TARGET_DEVICE=cpu ./practical
$ PRL_TARGET_DEVICE=acc ./practical
$ PRL_TARGET_DEVICE=1:0 ./practical
```

- Missing library `libprl_opencl.so` when launching?

```
--autorpath
```

Profiling

- Show profiling information at program exit

```
$ PRL_DUMP_ALL=1 ./practical
```

Profiling

- Show profiling information at program exit

```
$ PRL_DUMP_ALL=1 ./practical
```

- Log CPU<->GPU transfers and kernel executions

```
$ PRL_TRACE_GPU=1 ./practical
```

Benchmarking

```
typedef struct {
  float *gauss, waves, blurred;
  int32_t *hist;
} user_t;

static void init(void *arg) { ... }
static void kernel(void *arg) {
user_t *user = arg;
gen_waves(..., user->waves, ...);
...
}
static void finit(void *arg) { ... }

user_t user = { .gauss = gauss, ... };
prl_perf_benchmark(&kernel, &init, &finit, &user);
```

Invoke Benchmarking

- Measure total time

```
$ ./practical
```

Invoke Benchmarking

- Measure total time only

```
$ ./practical
```

- Measure CPU/GPU details

```
$ PRL_PROF_CPU=1 ./practical
$ PRL_PROF_GPU=1 ./practical
$ PRL_PROF_ALL=1 ./practical
```

Invoke Benchmarking

- Measure total time only

```
$ ./practical
```

- Measure CPU/GPU details

```
$ PRL_PROF_CPU=1 ./practical
$ PRL_PROF_GPU=1 ./practical
$ PRL_PROF_ALL=1 ./practical
```

- Control number of runs

```
$ PRL_PROF_RUNS=3 ./practical
$ PRL_PROF_DRY_RUNS=1 ./practical
```

Invoke Benchmarking

- Measure total time only

```
$ ./practical
```

- Measure CPU/GPU details

```
$ PRL_PROF_CPU=1 ./practical
$ PRL_PROF_GPU=1 ./practical
$ PRL_PROF_ALL=1 ./practical
```

- Control number of runs

```
$ PRL_PROF_RUNS=3 ./practical
$ PRL_PROF_DRY_RUNS=1 ./practical
```

- Add prefix to metrics (for easier regex matching)

```
$ PRL_PROF_PREFIX=My ./practical
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

- Print used size parameters

```
$ pencilcc practical.c ----dump-sizes
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

- Print used size parameters

```
$ pencilcc practical.c ----dump-sizes
```

- Per kernel setting

```
$ pencilcc practical.c --sizes="{kernel[0]->tile[16,4,4]}"
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

- Print used size parameters

```
$ pencilcc practical.c ----dump-sizes
```

- Per kernel setting

```
$ pencilcc practical.c --sizes="{kernel[0]->tile[16,4,4]}"
```

- Workgroup size

```
$ pencilcc practical.c --sizes="{kernel[i]->block[16,16,1]}"
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

- Print used size parameters

```
$ pencilcc practical.c ----dump-sizes
```

- Per kernel setting

```
$ pencilcc practical.c --sizes="{kernel[0]->tile[16,4,4]}"
```

- Workgroup size

```
$ pencilcc practical.c --sizes="{kernel[i]->block[16,16,1]}"
```

- Grid size

```
$ pencilcc practical.c --sizes="{kernel[i]->grid[512,512]}"
```

Optimization Parameters

- Tiling

```
$ pencilcc practical.c --tile-size=16
```

- Print used size parameters

```
$ pencilcc practical.c ----dump-sizes
```

- Per kernel setting

```
$ pencilcc practical.c --sizes="{kernel[0]->tile[16,4,4]}"
```

- Workgroup size

```
$ pencilcc practical.c --sizes="{kernel[i]->block[16,16,1]}"
```

- Grid size

```
$ pencilcc practical.c --sizes="{kernel[i]->grid[512,512]}"
```

- Combine sizes

```
--sizes="{kernel[i]->tile[16,4,4];kernel[i]->block[16,16,1]}"
```

Parameter Assumptions

- Some special cases do not happen

```
#include <pencil.h>

static void convolution(int count, int m, int n, ... \
                     int w, int h, ...) {
#pragma scop
__pencil_assume(w > 0);
__pencil_assume(h > 0);
__pencil_assume(m >= w);
__pencil_assume(n >= h);
```

Parameter Assumptions

- Some special cases do not happen

```
#include <pencil.h>

static void convolution(int count, int m, int n, ... \
                     int w, int h, ...) {
#pragma scop
__pencil_assume(w > 0);
__pencil_assume(h > 0);
__pencil_assume(m >= w);
__pencil_assume(n >= h);
```

- Limits for shared memory

```
__pencil_assume(w <= FILTER_WIDTH);
__pencil_assume(h <= FILTER_HEIGHT);
```

External Functions

- Insert OpenCL code: practical.cl

```
void hist_inc(__global int32_t *vec, int s) {
atomic_inc(&vec[s]);
}
```

External Functions

- Insert OpenCL code: practical.cl

```
void hist_inc(__global int32_t *vec, int s) {
atomic_inc(&vec[s]);
}
```

- What data does it access?

```
PENCIL_SUMMARY_FUNC void hist_inc_summary(int32_t *vec, int s){
PENCIL_USE(vec[s]);
}
void hist_inc(int32_t *vec, int s)
   PENCIL_ACCESS(hist_inc_summary);
void histogram(...) {
...
hist_inc(out[i], s);
```

External Functions

- Insert OpenCL code: practical.cl

```
void hist_inc(__global int32_t *vec, int s) {
atomic_inc(&vec[s]);
}
```

- What data does it access?

```
PENCIL_SUMMARY_FUNC void hist_inc_summary(int32_t *vec, int s){
PENCIL_USE(vec[s]);
}
void hist_inc(int32_t *vec, int s)
   PENCIL_ACCESS(hist_inc_summary);
void histogram(...) {
...
hist_inc(out[i], s);
```

- Include .cl into generated file

```
$ pencilcc practical.c --opencl-include-file=practical.cl
```

Force Parallelism

- Ignore loop-carried dependencies

```
#pragma pencil independent
for (int x = w/2; x < m-(h-1)/2; x += 1)
#pragma pencil independent
for (int y = w/2; y < n-(h-1)/2; y += 1)
```

Invalidate Data

- Invalidate data to avoid CPU->GPU transfers

```
static void convolution(..., float *out) {
#pragma scop
__pencil_kill(out)
...
```

Invalidate Data

- Invalidate data to avoid CPU->GPU transfers

```
static void convolution(..., float *out) {
#pragma scop
__pencil_kill(out)
...
```

- Invalidate data to avoid GPU->CPU transfers

```
void func(float tmp[static const restrict 8][8]) {
#pragma scop
tmp[?][?] = ...
use(tmp);
__pencil_kill(tmp)
#pragma endscop
}
```

Memory Allocation

- Reduce GPU memory allocation overhead

```
#include <prl.h>
...
float *waves = prl_alloc(N_IMAGES * ...);
...
prl_free(waves);
```

Memory Allocation

- Reduce GPU memory allocation overhead

```
#include <prl.h>
...
float *waves = prl_alloc(N_IMAGES * ...);
...
prl_free(waves);
```

- With option flags and handle

```
prl_mem waves_mem = prl_mem_alloc(N_IMAGES * ..., 0);
float *waves = prl_mem_get_host_mem(waves_mem);
...
prl_mem_free(waves_mem);
```

Buffer Flags

- Tell PRL that we never access the data on the CPU

```
prl_mem waves_mem = prl_mem_alloc(..., prl_mem_host_noaccess);
```

Buffer Flags

- Tell PRL that we never access the data on the CPU

```
prl_mem waves_mem = prl_mem_alloc(..., prl_mem_host_noaccess);
```

- Or never write it on the CPU

```
prl_mem hist_mem = prl_mem_alloc(..., prl_mem_host_nowrite);
```

Buffer Flags

- Tell PRL that we never access the data on the CPU

```
prl_mem waves_mem = prl_mem_alloc(..., prl_mem_host_noaccess);
```

- Or never write it on the CPU

```
prl_mem hist_mem = prl_mem_alloc(..., prl_mem_host_nowrite);
```

- Or not access it on the CPU from a certain point on

```
float *gauss = prl_mem_get_host_mem(gauss_mem);
init_gauss(..., gauss);
prl_mem_add_flags(gauss_mem, prl_mem_host_noaccess);
```

Afterword

- Project developed at `http://github.com/Meinersbur/pencilcc`
  - Still incomplete
  - Currently uses its own version of ppcg/pet/isl
  - Contributions/Bug Reports/Pull requests/etc welcome!
  - Mirror at `http::/github.com/pencil-language/pencilcc`

That's all Folks!